

Find: [Documents](#)[Citations](#)Searching for PHRASE **dynamically typed class declaration**.Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Google \(CiteSeer\)](#) [Google \(Web\)](#)
[CSB](#) [DBLP](#)No documents match Boolean query. Trying non-Boolean relevance query.
500 documents found. Order: relevance to query.

[Taming Message Passing: Efficient Method Look-Up for... - Vitek, Horspool \(1994\) \(Correct\) \(12 citations\)](#)
 Message Passing: Efficient Method Look-Up for **Dynamically Typed** Languages Jan Vitek 1 and R. Nigel
 Passing: Efficient Method Look-Up for **Dynamically Typed** Languages Jan Vitek 1 and R. Nigel Horspool 2
 selected on the basis of the message receiver's **class** and the message name (selector)Conceptually
cui.unige.ch/OSG/people/jvitek/Publications/ecoop94.ps.gz

[Representing Type Information in Dynamically Typed Languages - Gudeman \(1993\) \(Correct\) \(17 citations\)](#)
 Representing **Type** Information in **Dynamically Typed** Languages David Gudeman
 Representing **Type** Information in **Dynamically Typed** Languages David
ftp.cs.indiana.edu/pub/scheme-repository/doc/pubs/typeinfo.ps.gz

[Optimizing Dynamically-Typed Object-Oriented Languages... - Hölzle, Chambers, Ungar \(1991\) \(Correct\) \(9 citations\)](#)
 1 Optimizing **Dynamically-Typed** Object-Oriented Languages
 1 Optimizing **Dynamically-Typed** Object-Oriented Languages With Polymorphic Inline
self.sunlabs.com/papers/ecoop91.ps.Z

[Pickling State in the Java System - Riggs, Waldo, al. \(1996\) \(Correct\) \(33 citations\)](#)
 in the form of object **classes**. These can be **dynamically** loaded into a Java Virtual Machine, for
 to include meta information that identifies the **type** of each object and the relationships between
 of stateless computation in the form of object **classes**. In this paper we address the related task of
www.tns.lcs.mit.edu/~djw/library/coots96-riggs.ps.gz

[Style - A Practical Type Checker for Scheme - Lindig \(1993\) \(Correct\) \(3 citations\)](#)
 is used to **type** Scheme code. Although Scheme is **dynamically typed**, most parts of programs are statically
 Style A Practical **Type** Checker for Scheme Christian Lindig @
ftp.ips.cs.tu-bs.de/pub/local/softtech/papers/tr-93-10.ps.gz

[The Cecil Language, Specification and Rationale - Chambers \(1993\) \(Correct\) \(15 citations\)](#)
 Goals and Major Features 1 1.2 Overview 3 2 **Dynamically-Typed** Core .
 objectbased encapsulation, and optional static **type** checking. Cecil's static **type** system
cuiwww.unige.ch/OSG/people/jvitek/Resources/Archive/cecil-spec.ps.gz

[An Efficient Implementation of Self, a Dynamically-Typed... - Chambers, Ungar, Lee \(1991\) \(Correct\) \(26 citations\)](#)
 An Efficient Implementation of SELF, a **Dynamically-Typed** Object-Oriented Language Based on
 An Efficient Implementation of SELF, a **Dynamically-Typed** Object-Oriented Language Based on Prototypes
www.sunlabs.com/research/self/papers/coopsia89.ps.Z

[Formalizing the Eiffel Library Standard - Steven Atkinson \(1995\) \(Correct\)](#)
 Comment (all possible **class** and feature comments)**Type** (and **Class type**) all possible Eiffel (**class**)
 Eiffel language. It consists of a set of twenty-six **class** specifications, called the library kernel, which
www.csee.wvu.edu/~atkinson/pubs/tools18.ps.gz

[Implementing Haskell overloading - Augustsson \(1993\) \(Correct\) \(28 citations\)](#)
 have been advised to avoid it by using explicit **type** signatures. This is unfortunate since it does not
 way of handling overloading through the use of **type classes**. Haskell style **type classes** where introduced by
 A new **type class** is introduced by a **class declaration** which describes what the common operations
www.cs.chalmers.se/pub/cs-reports/papers/overload-fpca-93.ps.Z

[Don't Stop the BIBOP: Flexible and Efficient Storage... - Dybvig, Eby, Bruggeman \(1994\) \(Correct\) \(4 citations\)](#)
 Flexible and Efficient Storage Management for **Dynamically-Typed** Languages R. Kent Dybvig, David Eby, and
 and Efficient Storage Management for **Dynamically-Typed** Languages R. Kent Dybvig, David Eby, and Carl
ftp.cs.indiana.edu/pub/techreports/TR400.ps.Z

[The Impact of Structure Analysis on Prolog Compilation - Lindgren \(1996\) \(Correct\) \(1 citation\)](#)
 is potentially useful to reduce overheads in **dynamically typed** languages, such as Prolog or Lisp, where

structure analyzer for Prolog by extending previous **type** graph analyses, and evaluate its precision and
ftp.csd.uu.se/pub/papers/reports/0142.ps.gz

GJ Specification - Bracha, Odersky, Stoutamire, Wadler (1998) (Correct) (4 citations)

May 1998 1 Summary We propose to add generic **types** and methods to the Java TM programming
cm.bell-labs.com/cm/cs/who/wadler/topics/./papers/gj-specification/gj-specification-a4.ps.gz

An Object-Oriented Language System For Implementing Concurrent.. - Lucas (1993) (Correct) (1 citation)

case would be used for elements that have been **dynamically** allocated, the latter case would be used. The
5.2.1.3 Derived-**Type** Constructors with Additional Arguments

21 2.2.6 Derived **Classes**

www.best.com/~pjl/~pjl/resume/thesis.pdf

A Practical Soft Type System for Scheme - Wright, Cartwright (1994) (Correct) (57 citations)

assignment, and continuations. 1 Introduction **Dynamically typed** languages like Scheme [6] permit

A Practical Soft **Type** System for Scheme Andrew K. Wright Robert

www.cs.rice.edu/CS/PLT/Publications/./fp94-wc.ps.gz

Analysis of Objects with Dynamic and Multiple Inheritance - Agesen, Palsberg.. (1993) (Correct) (3 citations)

it allows the inheritance graph to change **dynamically**. Our algorithm handles this by deriving and
Springer Verlag Lecture Notes on Computer Science. **Type** Inference of SELF Analysis of Objects with
self.sunlabs.com/papers/ecoop93a.ps.Z

Message Dispatch on Pipelined Processors - Driesen, Hölzle, Vitek (1995) (Correct) (14 citations)

dispatch techniques for both statically- and **dynamically-typed** languages with both single and multiple
techniques for both statically- and **dynamically-typed** languages with both single and multiple
cul.unige.ch/OSG/people/jvitek/Publications/ecoop95.ps.gz

Simple objects for Standard ML - John Reppy (1996) (Correct) (22 citations)

use fall into one of two camps: either they are **dynamically typed**, and support very flexible uses of
a new approach to adding objects to the statically-**typed**, higher-order language Standard ML. Our approach
www.cs.wisc.edu/~cs536-1/submissions/141.ps.gz

OOLP: A Translation Approach to Object-Oriented Logic.. - Dalal, Gangopadhyay (1989) (Correct) (15 citations)

methods can be specified in a limited way. Two new **types** of method **declarations** are added -
interpretation of the object oriented concepts of **classes**, instances, messages and multiple inheritance.
describes the OOLP syntax of **class** and instance **declarations** and messages. In the next subsection, the
www.cs.columbia.edu/~dalal/papers/food89.ps.gz

Fast Interprocedural Class Analysis - DeFouw, Grove, Chambers (1997) (Correct) (50 citations)

the **classes** associated with the arguments to a **dynamically** dispatched message send call site determine
take O(N) time (such as Bacon and Sweeney's Rapid **Type** Analysis) We describe a general parameterized
Fast Interprocedural **Class** Analysis Greg DeFouw, David Grove, and Craig
ftp.cs.washington.edu/tr/1997/07/UW-CSE-97-07-02.PS.Z

Stack-Based Typed Assembly Language - Morrisett, Cray, Walker, Glew (1998) (Correct) (37 citations)

and software engineering advantages over their **dynamically typed** counterparts. Modern **type**-directed
Stack-Based **Typed** Assembly Language Greg Morrisett Karl Cray David
reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-178.ps

First 20 documents Next 20

Try your query at: [Google \(CiteSeer\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer - Copyright [NEC](#) and [IST](#)

An Efficient Implementation of Self, a Dynamically-Typed Object-Oriented Language Based on Prototypes (1991) ([Make Corrections](#)) ([32 citations](#))

Craig Chambers, David Ungar, Elgin Lee
SIGPLAN Notices



[Home/Search](#) [Bookmark](#) [Context](#) [Related](#)

View or download:

[sunlabs.com/research/se...oopsla89.ps.Z](#)

[sunlabs.com/papers/oopsla89.ps.Z](#)

Cached: [PS.gz](#) [PS](#) [PDF](#) [Image](#) [Update](#) [Help](#)

From: [sunlabs.com/research/self/pape...](#)
(more)

From: [sunlabs.com/pape...implementation](#)
(Enter author homepages)

([Enter summary](#))

Rate this article: 1 2 3 4 5 (best)
[Comment on this article](#)

Abstract: . We have developed and implemented techniques that double the performance of dynamically-typed object-oriented languages. Our SELF implementation runs twice as fast as the fastest Smalltalk implementation, despite SELF's lack of classes and explicit variables. To compensate for the absence of classes, our system uses implementation-level maps to transparently group objects cloned from the same prototype, providing data type information and eliminating the apparent space overhead for... ([Update](#))

Context of citations to this paper: [More](#)

.... is an operating system simulation that is used as a nontrivial program for evaluating performance of several object oriented languages [CHA 88] RNA is a parallel search program for predicting RNA secondary structures [NAK 95] This program uses an object to maintain and to...

...of how to ensure correct translation in the presence of multiple threads running on a shared memory multiprocessor. The Self compiler [8, 10] employs a specialization, called customization [9] that dynamically generates specialized versions of a method optimized for different...

Cited by: [More](#)

The Prototype-Instance Object Systems in - Amulet And Garnet ([Correct](#))

First Year Report - Harris (1998) ([Correct](#))

Implementation techniques for a multi-service Java Virtual Machine - Harris (1999) ([Correct](#))

Similar documents (at the sentence level):

9.7%: The Design and Implementation of the SELF Compiler, an.. - Chambers (1992) ([Correct](#))

Active bibliography (related documents): [More](#) [All](#)

0.7: Type-Checking and Type-Inference for Object-Oriented Programming.. - Graver (1989) ([Correct](#))

0.5: Adaptive Optimization For Self: Reconciling High Performance With .. - Hölzle (1994) ([Correct](#))

0.5: Iterative Type Analysis and Extended Message Splitting.. - Chambers, Ungar (1990) ([Correct](#))

System load high. Please wait...

Timeout. Please try your query later.

Similar documents based on text: [More](#) [All](#)

0.1: Outwitting GC Devils: A Hybrid Incremental Garbage Collector - Oopsla' Garbage Collection ([Correct](#))

0.1: Subjectivity in Object-Oriented Systems - Workshop Summary William ([Correct](#))

0.1: The Relativistic Composite-Velocity Reciprocity Principle - Ungar (2000) ([Correct](#))

Related documents from co-citation: [More](#) [All](#)

11: Making pure object-oriented languages practical - Chambers, Ungar - 1991

10: Efficient implementation of the Smalltalk-80 system - Deutsch, Schiffman - 1984

9: Optimizing dynamicallydispatched calls with run-time type feedback (context) - Holzle, Ungar - 1994

BibTeX entry: ([Update](#))

Craig Chambers, David Ungar, and Elgin Lee, "An Efficient Implementation of SELF, a DynamicallyTyped Object-Oriented Language Based on Prototypes." In OOPSLA '89 Conference Proceedings, pp. 49-70, New Orleans, LA, 1989. Published as SIGPLAN Notices 24(10), October, 1989. <http://citeseer.ist.psu.edu/14611.html> [More](#)

```
@inproceedings{ chambers89efficient,
  author = "C. Chambers and D. Ungar and E. Lee",
  title = "An Efficient Implementation of {SELF} a Dynamically-Typed Object-Oriented Lang",
  booktitle = "Proceedings of the Conference on Object-Oriented Programming Systems, Lang",
  journal = "SIGPLAN Notices",
```

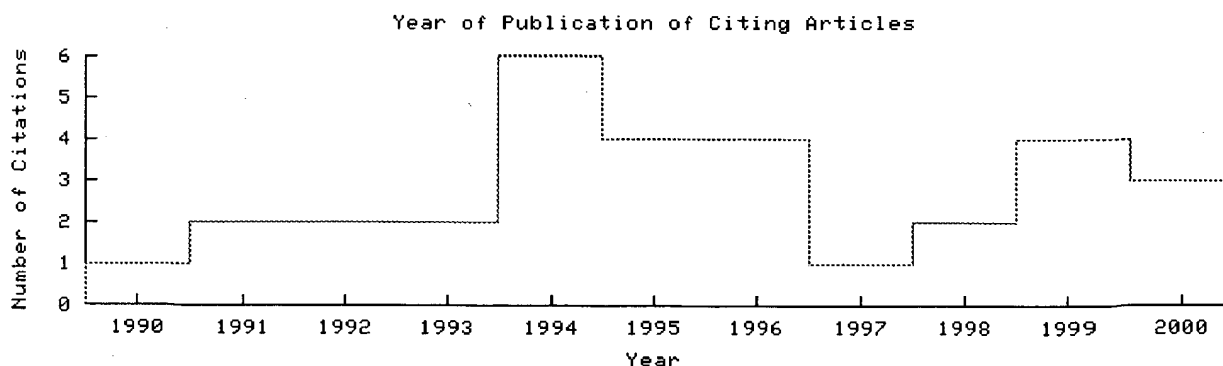
```

volume = "24",
number = "10",
publisher = "ACM Press",
address = "New York, NY",
editor = "Norman Meyrowitz",
pages = "49--70",
year = "1989",
url = "citeseer.ist.psu.edu/14611.html" )

```

Citations (may not include all citations):

- 960 Programming Language (context) - Stroustrup, The - 1986
- 470 Smalltalk-80: The Language and its Implementation (context) - Goldberg, Robson - 1983
- 163 Dimensions of Object-Based Language Design (context) - Wegner - 1987
- 137 Customization: Optimizing Compiler Technology for SELF (context) - Chambers, Ungar - 1989
- 135 Using Prototypical Objects to Implement Shared Behavior in O.. - Lieberman - 1986
- 112 Efficient Implementation of the Smalltalk -80 System - Deutsch, Schiffman - 1984
- 100 Common Lisp Object System Specification (context) - Bobrow, DeMichiel et al. - 1988
- 92 An Introduction to Trellis/Owl (context) - Schaffert, Cooper et al. - 1986
- 91 Object-Oriented Programming with Flavors (context) - Moon - 1986
- 67 Delegation is Inheritance (context) - Stein - 1987
- 44 Classes Versus Prototypes in Object-Oriented Languages (context) - Borning - 1986
- 41 Genericity versus Inheritance (context) - Meyer - 1986
- 41 Tenuring Policies for Generation-Based Storage Reclamation (context) - Ungar, Jackson - 1988
- 39 An Exemplar Based Smalltalk (context) - LaLonde, Thomas et al. - 1986
- 34 TS: An Optimizing Compiler for Smalltalk (context) - Johnson, Graver et al. - 1988
- 29 Inferring Types in Smalltalk (context) - Suzuki - 1981
- 29 A type declaration and inference system for Smalltalk (context) - Borning, Ingalls - 1982
- 24 MIT Artificial Intelligence Laboratory (context) - Steele, Jr - 1976
- 24 MIT Artificial Intelligence Laboratory (context) - Steele - 1976
- 9 Hurricane: An Optimizing Compiler for Smalltalk (context) - Atkinson - 1986
- 7 a Prototype-Based ObjectOriented Programming Language (context) - Lee - 1988
- 6 The Dorado Smalltalk-80 Implementation: Hardware Architectur.. (context) - Deutsch - 1983
- 5 The Smalltalk-80 Benchmarks (context) - McCall - 1983
- 4 Personal communication (context) - Deutsch - 1988
- 3 Stanford Integer benchmarks (context) - Hennessy - 1988
- 2 QUICKTALK: A Smalltalk80 Dialect for Defining Primitive Meth.. (context) - Ballard, Maier et al. - 1986
- 1 ParcPlace Newsletter (context) - Systems - 1988
- 1 Type inferencing in Smalltalk (context) - Curtis - 1989
- 1 CMOS Gate Array Implementation of the SPARC Architecture (context) - Namjoo, Agrawal et al. - 1988



The graph only includes citing articles where the year of publication is known.

Documents on the same site (<http://www.sunlabs.com/research/self/papers/>): [More](#)

Experiencing - Objects An (Correct)

Organizing Programs Without Classes - Ungar, Chambers, Chang, Hölzle (1991) (Correct)

Integrating Independently-Developed Components in Object-Oriented .. - Hölzle (1993) (Correct)

[Online articles have much greater impact](#) [More about CiteSeer.IST](#) [Add search form to your site](#) [Submit documents](#)
[Feedback](#)



Subscribe Register Login
(Full Service) (Limited Service, Free)

Search: ☒ The ACM Digital Library ☐ The Guide
Efficient Implementation of SELF <and> Chambers

THE ACM DIGITAL LIBRARY

Feedback

Terms used Efficient Implementation of SELF and Chambers

Sort results
by

relevance

☒ Save results to a Binder

Try

☒ Search Tips

Try

☐ Open results in a new window

Display results

expanded form

Results 1 - 20 of 200

Result page: **1** 2 3 4 5 6 7 8 9

Best 200 shown

1 A third-generation SELF implementation: reconciling responsiveness with p

Urs Hölzle, David Ungar

October 1994 ACM SIGPLAN Notices , Proceedings of the ninth annual conference
systems, language, and applications, Volume 29 Issue 10

Full text available: pdf(1.89 MB)

Additional Information: full citation, abstract, references, ci

Programming systems should be both responsive (to support rapid developme
computations quickly). Pure object-oriented languages are harder to impleme
optimization to achieve good performance. Unfortunately, optimization conflic
because it tends to produce long compilation pauses, leading to unresponsive
to achieve good responsiveness, existing exploratory progra ...

2 Kaleidoscope: mixing objects, constraints, and imperative programming

Bjorn N. Freeman-Benson

September 1990 ACM SIGPLAN Notices , Proceedings of the European conference o
Object-oriented programming systems, languages, and applicatio

Full text available: pdf(1.14 MB)


Additional Information: full citation, abstract, references, ci

Kaleidoscope is an object-oriented language being designed to integrate the ti
paradigm with the less traditional declarative constraint paradigm. Imperative
declarative constraints provide object relations. A variables as streams seman
integration. A running example is used to illustrate the language concepts&mc

3 Selective specialization for object-oriented languages

Jeffrey Dean, Craig Chambers, David Grove

June 1995 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 conference and implementation, Volume 30 Issue 6

Full text available:  pdf(1.32 MB)


Additional Information: full citation, abstract, references, ci

Dynamic dispatching is a major source of run-time overhead in object-oriented languages. It is due to the indirect effect of preventing other optimizations. Compilers for object-oriented languages analyze the classes of objects stored in memory, bounding the possible classes of message receivers enough so that the compiler can generate a message send at compile time ...

4 Optimizing dynamically-dispatched calls with run-time type feedback

Urs Hölzle, David Ungar

June 1994 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference and implementation, Volume 29 Issue 6


Full text available:  pdf(1.39 MB)

Additional Information: full citation, references, citations, inc

5 Debugging optimized code with dynamic deoptimization

Urs Hölzle, Craig Chambers, David Ungar

July 1992 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1992 conference and implementation, Volume 27 Issue 7

Full text available:  pdf(1.26 MB)


Additional Information: full citation, abstract, references, ci

SELF's debugging system provides complete source-level debugging (expected). It shields the debugger from optimizations performed by the compiler by dynamic deoptimization. Deoptimization only affects the procedure activations that are actively being compiled. Deoptimization requires the compiler to supply debugging information.

6 An efficient implementation of SELF a dynamically-typed object-oriented language

C. Chambers, D. Ungar, E. Lee

September 1989 ACM SIGPLAN Notices , Conference proceedings on Object-oriented languages, Volume 24 Issue 10

Full text available:  pdf(2.41 MB)


Additional Information: full citation, abstract, references, ci

We have developed and implemented techniques that double the performance of object-oriented languages. Our SELF implementation runs twice as fast as the fastest Smalltalk implementation. It has no classes and explicit variables. To compensate for the absence of classes, our implementation maps to transparently group objects cloned from the same prototype, providing the apparent ...





7 Making pure object-oriented languages practical

Craig Chambers, David Ungar

November 1991 ACM SIGPLAN Notices , Conference proceedings on Object-oriented languages, Volume 26 Issue 11

Full text available:  pdf(1.86 MB)


Additional Information: full citation, references, citations

- 8 **A framework for selective recompilation in the presence of complex interm**
Craig Chambers, Jeffrey Dean, David Grove
April 1995 Proceedings of the 17th international conference on Software engine
Full text available:  pdf(1.12 MB) Additional Information: full citation, references, citings, index te
- 9 **Type feedback vs. concrete type inference: a comparison of optimization te**
languages
Ole Agesen, Urs Hölzle
October 1995 ACM SIGPLAN Notices , Proceedings of the tenth annual conference
systems, languages, and applications, Volume 30 Issue 10
Full text available:  pdf(2.27 MB) Additional Information: full citation, abstract, references, ci
Two promising optimization techniques for object-oriented languages are type
prediction) and concrete type inference (static analysis). We directly compare
effectiveness on a suite of 23 SELF programs while keeping other factors cons
inline over 95% of all sends and deliver similar overall performance with one
machine integer ...
- 10 **Prototype-based languages: from a new taxonomy to constructive proposal**
Christophe Dony, Jacques Malenfant, Pierre Cointe
October 1992 ACM SIGPLAN Notices , conference proceedings on Object-oriented
applications, Volume 27 Issue 10
Full text available:  pdf(2.17 MB) Additional Information: full citation, references, citings
- 11 **The impact of interprocedural class analysis on optimization**
David Grove
November 1995 Proceedings of the 1995 conference of the Centre for Advanced S
Full text available:  pdf(55.52 KB) Additional Information: full citation, abstract, references, c
The runtime performance of object-oriented languages often suffers due to th
order to make these languages competitive with traditional languages, optimi
many of the dynamic dispatches as possible. A variety of local and intraproce
do this, but they can be ineffective when they are unable to statically bind an
better analysis across noninlined messag ...

12 Shade: a fast instruction-set simulator for execution profiling

Bob Cmelik, David Keppel

May 1994 ACM SIGMETRICS Performance Evaluation Review , Proceedings of the Measurement and modeling of computer systems, Volume 22 Issue 1

Full text available:  pdf(1.28 MB)


Additional Information: full citation, abstract, references, citi

Tracing tools are used widely to help analyze, design, and tune both hardware describes a tool called Shade which combines efficient instruction-set simulati generation capability. Efficiency is achieved by dynamically compiling and cac application program. The user may control the extent of tracing in a variety of state information may be collected ...

13 Call forwarding: a simple interprocedural optimization technique for dynami

Koen De Bosschere, Saumya Debray, David Gudeman, Sampath Kannan

February 1994 Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Princ

Full text available:  pdf(1.16 MB)

Additional Information: full citation, abstract, references, citi

This paper discusses call forwarding, a simple interprocedural optimization tec languages. The basic idea behind the optimization is straightforward: find an actions” of a procedure, and generate multiple entry points for the pro realized from different call sites bypassing different sets of entry actions. We : optimal solutions to arbitrary ...

14 Declarative programming in a prototype-instance system: object-oriented p

Brad A. Myers, Dario A. Giuse, Brad Vander Zanden

October 1992 ACM SIGPLAN Notices , conference proceedings on Object-oriented , applications, Volume 27 Issue 10

Full text available:  pdf(2.19 MB)

Additional Information: full citation, references, citings

15 Representation of medical guidelines using a classification-based system

C. Heinlein, K. Kuhn, P. Dadam

November 1994 Proceedings of the third international conference on Information :

Full text available:  pdf(872.92 KB)

Additional Information: full citation, abstract, references

Medical guidelines play an increasing role in selecting diagnostic and therapeutic effectiveness, invasiveness, and costs. To work directly on patient data already should be integrated into a medical information system. In order to develop a module” (MGM) managing and applying guidelines to patients, a &ldqu representation of guidelines is necessary which reflects the structu ...

16 Software components in a data structure precompiler

Marty Sirkin, Don Batory, Vivek Singhal

May 1993 Proceedings of the 15th international conference on Software Engineeri

Full text available:  pdf(965.99 KB)

Additional Information: full citation, references, citings

17 Java intermediate bytecodes: ACM SIGPLAN workshop on intermediate re

James Gosling

March 1995 ACM SIGPLAN Notices , Papers from the 1995 ACM SIGPLAN workshop
Volume 30 Issue 3

Full text available:  pdf(515.70 KB)

Additional Information: full citation, abstract, citing

Java is a programming language loosely related to C++. Java originated in a development environment for small distributed embedded systems. Programs “safe” and portable. These needs led to a design that is rather d particular, the form of compiled programs is machine independent bytecodes. in ways usually associated with higher level, more abstract inte ...

18 Implementing Haskell overloading

Lennart Augustsson

July 1993 Proceedings of the conference on Functional programming languages a


Full text available:  pdf(820.14 KB)

Additional Information: full citation, references, citings, inde

19 Profile-guided receiver class prediction

David Grove, Jeffrey Dean, Charles Garrett, Craig Chambers

October 1995 ACM SIGPLAN Notices , Proceedings of the tenth annual conference
systems, languages, and applications, Volume 30 Issue 10

Full text available:  pdf(1.78 MB)

Additional Information: full citation, abstract, references, ci

The use of dynamically-dispatched procedure calls is a key mechanism for wri object-oriented languages. Unfortunately, dynamic dispatching imposes a runt implementations of pure object-oriented languages have utilized profile-guide performance penalty, and some researchers have argued for applying receiver like C++. We performed a detailed examina ...

20 Vortex: an optimizing compiler for object-oriented languages

Jeffrey Dean, Greg DeFouw, David Grove, Vassily Litvinov, Craig Chambers

October 1996 ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN confe
systems, languages, and applications, Volume 31 Issue 10

Full text available:  pdf(2.45 MB)

Additional Information: full citation, abstract, references, ci

Previously, techniques such as class hierarchy analysis and profile-guided rec demonstrated to greatly improve the performance of applications written in p degree to which these results are transferable to applications written in hybri answer this question, we have developed the Vortex compiler infrastructure, a compiler for object-oriented languages, with ...

Results 1 - 20 of 200

Result page: **1** 2 3 4 5 6 7

The ACM Portal is published by the Association for Computing Machinery. C

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Cont](#)

Find: Searching for PHRASE **dynamically typed program declaration**.Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Google \(CiteSeer\)](#) [Google \(Web\)](#)
[CSB](#) [DBLP](#)

No documents match Boolean query. Trying non-Boolean relevance query.

500 documents found. Order: relevance to query.

[Representing Type Information in Dynamically Typed Languages - Gudeman \(1993\) \(Correct\) \(17 citations\)](#)Representing **Type** Information in **Dynamically Typed** Languages David GudemanRepresenting **Type** Information in **Dynamically Typed** Languages David<ftp.cs.indiana.edu/pub/scheme-repository/doc/pubs/typeinfo.ps.gz>[Taming Message Passing: Efficient Method Look-Up for... - Vitek, Horspool \(1994\) \(Correct\) \(12 citations\)](#)Message Passing: Efficient Method Look-Up for **Dynamically Typed** Languages Jan Vitek 1 and R. NigelPassing: Efficient Method Look-Up for **Dynamically Typed** Languages Jan Vitek 1 and R. Nigel Horspool 2Message passing is at the heart of object-oriented **programming** the efficiency of this mechanism is crucial<cui.unige.ch/OSG/people/jvitek/Publications/ecoop94.ps.gz>[Optimizing Dynamically-Typed Object-Oriented Languages... - Hölzle, Chambers, Ungar \(1991\) \(Correct\) \(9 citations\)](#)1 Optimizing **Dynamically-Typed** Object-Oriented Languages1 Optimizing **Dynamically-Typed** Object-Oriented Languages With Polymorphic Inline<self.sunlabs.com/papers/ecoop91.ps.Z>[The Cecil Language, Specification and Rationale - Chambers \(1993\) \(Correct\) \(15 citations\)](#)Goals and Major Features 1 1.2 Overview 3 2 **Dynamically-Typed** Coreobjectbased encapsulation, and optional static **type** checking. Cecil's static **type** system<cuiwww.unige.ch/OSG/people/jvitek/Resources/Archive/cecil-spec.ps.gz>[Style - A Practical Type Checker for Scheme - Lindig \(1993\) \(Correct\) \(3 citations\)](#)is used to **type** Scheme code. Although Scheme is **dynamically typed**, most parts of **programs** are staticallyStyle A Practical **Type** Checker for Scheme Christian Lindig @<ftp.lps.cs.tu-bs.de/pub/local/softtech/papers/tr-93-10.ps.gz>[An Efficient Implementation of Self, a Dynamically-Typed... - Chambers, Ungar, Lee \(1991\) \(Correct\) \(26 citations\)](#)An Efficient Implementation of SELF, a **Dynamically-Typed** Object-Oriented Language Based onAn Efficient Implementation of SELF, a **Dynamically-Typed** Object-Oriented Language Based on Prototypes<www.sunlabs.com/research/self/papers/copsia89.ps.Z>[Don't Stop the BIBOP: Flexible and Efficient Storage... - Dybvig, Eby, Bruggeman \(1994\) \(Correct\) \(4 citations\)](#)Flexible and Efficient Storage Management for **Dynamically-Typed** Languages R. Kent Dybvig, David Eby, andand Efficient Storage Management for **Dynamically-Typed** Languages R. Kent Dybvig, David Eby, and Carl<ftp.cs.indiana.edu/pub/techreports/TR400.ps.Z>[On the Runtime Complexity of Type-Directed Unboxing - Minamide \(1998\) \(Correct\) \(4 citations\)](#)cannot be completely known at compile time. In **dynamically typed** languages like Lisp, the basic approachOn the Runtime Complexity of **Type**-Directed Unboxing Yasuhiko Minamidewww.score.is.tsukuba.ac.jp/~minamide/unboxing_icfp98.ps.gz[The Impact of Structure Analysis on Prolog Compilation - Lindgren \(1996\) \(Correct\) \(1 citation\)](#)is potentially useful to reduce overheads in **dynamically typed** languages, such as Prolog or Lisp, wherestructure analyzer for Prolog by extending previous **type** graph analyses, and evaluate its precision and<ftp.csd.uu.se/pub/papers/reports/0142.ps.gz>[A Practical Soft Type System for Scheme - Wright, Cartwright \(1994\) \(Correct\) \(57 citations\)](#)assignment, and continuations. 1 Introduction **Dynamically typed** languages like Scheme [6] permitA Practical Soft **Type** System for Scheme Andrew K. Wright Robert<www.cs.rice.edu/CS/PLT/Publications/./fp94-wc.ps.gz>[Dynamic Typing in a Statically Typed Language - Abadi, Cardelli, Pierce, Plotkin \(1989\) \(Correct\) \(106 citations\)](#)the **type** Dynamic. We give examples of how **dynamically typed** values can be used in **programming**. ThenDynamic **Typing** in a Statically **Typed** Language Mart'in Abadi<www.cs.indiana.edu/www/pub/pierce/dynamic.ps.gz>

[A Typed Intermediate Language for Flow-Directed Compilation - Wells, Dimock, Muller.. \(1997\) \(Correct\) \(13 citations\)](#)

A Typed Intermediate Language for Flow-Directed

www.cs.bu.edu/groups/church/reports/typed-IL-for-flow-directed-compilation.ps.gz

[Message Dispatch on Pipelined Processors - Driesen, Hölzle, Vitek \(1995\) \(Correct\) \(14 citations\)](#)

dispatch techniques for both statically- and **dynamically-typed** languages with both single and multiple techniques for both statically- and **dynamically-typed** languages with both single and multiple

cui.unige.ch/OSG/people/vitek/Publications/ecoop95.ps.gz

[Simple objects for Standard ML - John Reppy \(1996\) \(Correct\) \(22 citations\)](#)

use fall into one of two camps: either they are **dynamically typed**, and support very flexible uses of a new approach to adding objects to the statically-**typed**, higher-order language Standard ML. Our approach

www.cs.wisc.edu/~cs536-1/submissions/141.ps.gz

[Fast Interprocedural Class Analysis - DeFouw, Grove, Chambers \(1997\) \(Correct\) \(50 citations\)](#)

the classes associated with the arguments to a **dynamically** dispatched message send call site determine take O(N) time (such as Bacon and Sweeney's Rapid **Type** Analysis)We describe a general parameterized on a collection of substantial Cecil and Java **programs**. 1 Introduction Interprocedural class analysis

ftp.cs.washington.edu/tr/1997/07/UW-CSE-97-07-02.PS.Z

[The Rthreads Distributed Shared Memory System - Dreier, Zahn, Ungerer \(1998\) \(Correct\) \(1 citation\)](#)

is placed in a buffer space that is allocated **dynamically** or on memory pages in the case of page-based a **program** running on these systems contains two **types** of global data: Global data shared with other way, because all global variables of an Rthreads **program** are shared among the participating nodes and are

goethe.ira.uka.de/people/ungerer/Rthreads-Colorado.ps

[Stack-Based Typed Assembly Language - Morrisett, Cray, Walker, Glew \(1998\) \(Correct\) \(37 citations\)](#)

and software engineering advantages over their **dynamically typed** counterparts. Modern **type**-directed

Stack-Based Typed Assembly Language Greg Morrisett Karl Cray David

reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-178.ps

[A Practical Approach to Type Inference for EuLisp - Andreas Kind \(1993\) \(Correct\) \(10 citations\)](#)

Drawbacks in efficiency, apparent in Lisp as a **dynamically typed programming** language, can be avoided by In The Netherlands A Practical Approach To **Type** Inference For Eulisp Andreas Kind

www.maths.bath.ac.uk/~ak1/papers/infer.ps

[Analysis of Objects with Dynamic and Multiple Inheritance - Agesen, Palsberg.. \(1993\) \(Correct\) \(3 citations\)](#)

it allows the inheritance graph to change **dynamically**. Our algorithm handles this by deriving and Springer Verlag Lecture Notes on Computer Science. **Type** Inference of SELF Analysis of Objects with

self.sunlabs.com/papers/ecoop93a.ps.Z

[Practical Refinement-Type Checking - Davies \(1997\) \(Correct\) \(9 citations\)](#)

modularity, particularly when the language is **dynamically typed** [FFK 96]Unfortunately, these Thesis Proposal: Practical Refinement-**type** Checking Rowan Davies November 13, 1997 Abstract

www.cs.cmu.edu/afs/cs/user/rowan/www/papers/proposal.ps

[First 20 documents](#) [Next 20](#)

Try your query at: [Google \(CiteSeer\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer.IST - Copyright [NEC](#) and [IST](#)

Find: [Documents](#)[Citations](#)Searching for **dynamically typed and late dynamic binding**.Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Google \(CiteSeer\)](#) [Google \(Web\)](#)
[CSB](#) [DBLP](#)

No documents match Boolean query. Trying non-Boolean relevance query.

500 documents found. **Order: relevance to query.**

[Taming Message Passing: Efficient Method Look-Up for... - Vitek, Horspool \(1994\) \(Correct\) \(12 citations\)](#)
[Message Passing: Efficient Method Look-Up for Dynamically Typed Languages Jan Vitek 1 and R. Nigel](#)
[Passing: Efficient Method Look-Up for Dynamically Typed Languages Jan Vitek 1 and R. Nigel Horspool 2](#)
 a bad inlining decision in a base class can prevent later redefinitions of methods in derived classes. On
cui.unige.ch/OSG/people/jvitek/Publications/ecoop94.ps.gz

[A Syntactic Theory of Dynamic Binding - Moreau \(1997\) \(Correct\) \(1 citation\)](#)
 duration of an expression, i.e. the handler is **dynamically** bound during the extent of the expression.
 Uses of **Dynamic Binding** 2.1 Conciseness A **typical** use of **dynamic binding** is a printing routine
 of an expression. A **dynamic** variable refers to the **latest** active **dynamic binding** that exists for that
www.mmrg.ecs.soton.ac.uk/publications/archive/moreau1997e/moreau1997e.ps

[A Lambda-Calculus for Dynamic Binding - Dami \(1997\) \(Correct\) \(18 citations\)](#)
 required to be increasingly "open" able to **dynamically** interact with other, possibly unknown or
 properties of the -calculus, has a Curry-style **type** inference system, and has a formal notion of
 stacks of dictionaries in FORTH or Postscript, **late binding** of message names to Preprint submitted to
cui.unige.ch/pub/dami/dynBind.ps.Z

[Object-Oriented Type Inference - Palsberg, Schwartzbach \(1991\) \(Correct\) \(121 citations\)](#)
Late binding means that a message send is **dynamically** bound to an implementation depending on the
 Object-Oriented **Type** Inference Jens Palsberg and Michael I.
ftp.daimi.aau.dk/pub/palsberg/papers/oopsla91.ps.Z

[Representing Type Information in Dynamically Typed Languages - Gudeman \(1993\) \(Correct\) \(17 citations\)](#)
[Representing Type Information in Dynamically Typed Languages David Gudeman](#)
[Representing Type Information in Dynamically Typed Languages David](#)
ftp.cs.indiana.edu/pub/scheme-repository/doc/pubs/typeinfo.ps.gz

[A Configurable Protocol Architecture for CORBA Environments - Crane, Dulay \(1997\) \(Correct\) \(6 citations\)](#)
 to select the elements of a protocol stack **dynamically** at **bind-time** depending on the properties of
 multiple styles of **binding** between different **types** of interaction classes over multiple transport
 is two-way. A client issues a request and at some **later** time, a server returns one or more replies. Since
scorch.doc.ic.ac.uk/~jsc/research/isads-97.ps.gz

[Optimizing Dynamically-Typed Object-Oriented Languages.. - Hölzle, Chambers, Ungar \(1991\) \(Correct\) \(9 citations\)](#)
 1 Optimizing **Dynamically-Typed** Object-Oriented Languages
 1 Optimizing **Dynamically-Typed** Object-Oriented Languages With Polymorphic Inline
self.sunlabs.com/papers/ecoop91.ps.Z

[Using Visualization to Foster Object-Oriented Program... - Jerding, Stasko \(1994\) \(Correct\) \(10 citations\)](#)
 these issues-many of which arise at run-time. **Typically**, a programmer develops a computer program to
 a means for characterizing both static and **dynamic** aspects of an object-oriented program. We then
 programmers must understand inheritance, **dynamic binding**, and various forms of polymorphism. A software
ftp.cc.gatech.edu/pub/gvu/tr/1994/94-33.ps.Z

[An Analysis of pi-calculus Bisimulations - Amadio, Ait-Mohamed \(1995\) \(Correct\)](#)
 for instance it allows to model networks with a **dynamically** changing topology [17]and reasonable
 kind of formalization, inspired by previous work in **type** theory, is a useful bridge between intuitive
 (3) The introduction of a refinement of the '**late**' bisimulation which we call 'uniform'Roughly the
prolis.univ-mrs.fr/~amadio/pianalysis93.ps.gz

[The Cecil Language, Specification and Rationale - Chambers \(1993\) \(Correct\) \(15 citations\)](#)
 Goals and Major Features 1 1.2 Overview 3 2 **Dynamically-Typed** Core .
 objectbased encapsulation, and optional static **type** checking. Cecil's static **type** system
cuiwww.unige.ch/OSG/people/jvitek/Resources/Archive/cecil-spec.ps.gz

Dynamic Typing in a Statically Typed Language - Abadi, Cardelli, Pierce, Plotkin (1989) (Correct) (106 citations)
 the **type Dynamic**. We give examples of how **dynamically typed** values can be used in programming. Then
Dynamic Typing in a Statically Typed Language Mart'in Abadi
www.cs.indiana.edu/~www/pub/pierce/dynamic.ps.gz

Style - A Practical Type Checker for Scheme - Lindig (1993) (Correct) (3 citations)
 is used to **type** Scheme code. Although Scheme is **dynamically typed**, most parts of programs are statically
Style A Practical Type Checker for Scheme Christian Lindig @
ftp.ips.cs.tu-bs.de/pub/local/softtech/papers/tr-93-10.ps.gz

Message Dispatch on Pipelined Processors - Driesen, Hölzle, Vitek (1995) (Correct) (14 citations)
 dispatch techniques for both statically- and **dynamically-typed** languages with both single and multiple
 techniques for both statically- and **dynamically-typed** languages with both single and multiple
cul.unige.ch/OSG/people/vitek/Publications/ecoop95.ps.gz

Analysis of Objects with Dynamic and Multiple Inheritance - Agesen, Palsberg.. (1993) (Correct) (3 citations)
 it allows the inheritance graph to change **dynamically**. Our algorithm handles this by deriving and
 Springer Verlag Lecture Notes on Computer Science. **Type Inference of SELF** Analysis of Objects with
self.sunlabs.com/papers/ecoop93a.ps.Z

Visual Segmentation and the Dynamic Binding Problem: Improving the ... - Smith (1993) (Correct)
 enumerated representations. Enumeration does not **dynamically bind** representational elements together at
 now follows a discussion of **dynamic binding** and the **types** of connectionist representations which have been
 Report: Ct92-15-003 Visual Segmentation And The **Dynamic Binding** Problem Improving The Robustness Of An
www.cis.plym.ac.uk/MAST_Reports/ct92-15-003.ps.gz

An Efficient Implementation of Self, a Dynamically-Typed... - Chambers, Ungar, Lee (1991) (Correct) (26 citations)
 An Efficient Implementation of SELF, a **Dynamically-Typed** Object-Oriented Language Based on
 An Efficient Implementation of SELF, a **Dynamically-Typed** Object-Oriented Language Based on Prototypes
www.sunlabs.com/research/self/papers/copsia89.ps.Z

Compact Dispatch Tables for Dynamically Typed Object Oriented ... - Vitek, Horspool (Correct) (8 citations)
 Compact Dispatch Tables for **Dynamically Typed** Object Oriented Languages Abstract.
 Compact Dispatch Tables for **Dynamically Typed** Object Oriented Languages Abstract. **Dynamically**
 at run-time. This is called **dynamic binding** or **late binding**. In object-oriented programming, **dynamic**
www.csr.uvic.ca/~nigelh/Publications/cdt95.pdf

Dynamic Lambda Calculus - Kohlhase, Kuschert (1997) (Correct) (4 citations)
 modeg. Thus, only those variables which are not **dynamically** bound in the argument of the negation, i.e.
 discourse semantics as an analogon to the simply **typed** -calculus. Just as that can be specialized to
kbibmp3.ub.uni-kl.de/Preprint_Informatik/PS/KoKu97-Claus.ps.gz

M-RPC: A Remote Procedure Call Service for Mobile Clients - Bakre, Badrinath (1995) (Correct) (5 citations)
 applications will also need the ability to **dynamically bind** mobile clients to local information
 traffic. The client side of such an application typically gets the server address using a name lookup
 to handle switchable server **bindings** are described later. 7. The agent at the new MSR starts a worker
paul.rutgers.edu/pub/badri/mrpc.ps.Z

First 20 documents Next 20

Try your query at: [Google \(CiteSeer\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer.IST - Copyright [NEC](#) and [IST](#)

Find: [Documents](#)[Citations](#)Searching for **ibm and late dynamic binding**.Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Google \(CiteSeer\)](#) [Google \(Web\)](#)
[CSB](#) [DBLP](#)

No documents match Boolean query. Trying non-Boolean relevance query.

500 documents found. **Order: relevance to query.**[A Syntactic Theory of Dynamic Binding - Moreau \(1997\) \(Correct\) \(1 citation\)](#)of an expression. A **dynamic** variable refers to the **latest active dynamic binding** that exists for thatA Syntactic Theory of **Dynamic Binding** To appear in the proceedings ofwww.mmrq.ecs.soton.ac.uk/publications/archive/moreau1997e/moreau1997e.ps[A Lambda-Calculus for Dynamic Binding - Dami \(1997\) \(Correct\) \(18 citations\)](#)stacks of dictionaries in FORTH or Postscript, **late binding** of message names to Preprint submitted toA Lambda-Calculus for **Dynamic Binding** Laurent Dami Centre UniversitaireGenève 4, Switzerland dami@cul.unige.ch **Dynamic binding** is a runtime lookup operation which extractscul.unige.ch/pub/dami/dynBind.ps.Z[Using Visualization to Foster Object-Oriented Program... - Jerding, Stasko \(1994\) \(Correct\) \(10 citations\)](#)

mapping from visualization model to view, while the IBM system can have a many-to-one mapping of models to

a means for characterizing both static and **dynamic** aspects of an object-oriented program. We thenprogrammers must understand inheritance, **dynamic binding**, and various forms of polymorphism. A softwareftp.cc.gatech.edu/pub/gvu/tr/1994/94-33.ps.Z[Object-Oriented Type Inference - Palsberg, Schwartzbach \(1991\) \(Correct\) \(121 citations\)](#)programs with inheritance, assignments, and **late binding**. It guarantees that all messages are**Binding Late binding** means that a message send is **ynamically** bound to an implementation depending on theprograms with inheritance, assignments, and **late binding**. It guarantees that all messages are understood,ftp.daimi.aau.dk/pub/palsberg/papers/eopsla91.ps.Z[Visual Segmentation and the Dynamic Binding Problem: Improving the ... - Smith \(1993\) \(Correct\)](#)Report: Ct92-15-003 Visual Segmentation And The **Dynamic Binding** Problem Improving The Robustness Of AnCt92-15-003 Visual Segmentation And The **Dynamic Binding** Problem Improving The Robustness Of Ansuch a device requires the solution of the **dynamic binding** problem. Visual segmentation could be learned bywww.cis.plym.ac.uk/MAST/_Reports/ct92-15-003.ps.gz[An Analysis of pi-calculus Bisimulations - Amadio, Ait-Mohamed \(1995\) \(Correct\)](#)(3) The introduction of a refinement of the '**late**' bisimulation which we call 'uniform' Roughly thesimple framework in which the semantics of the **dynamic** creation and transmission of channels can be3)P Q iff P Q Intermezzo: early and **late binding**. What synchronization tree should one associateprolis.univ-mrs.fr/~amadio/planalysis93.ps.gz[Taming Message Passing: Efficient Method Look-Up for... - Vitek, Horspool \(1994\) \(Correct\) \(12 citations\)](#)a bad inlining decision in a base class can prevent **later** redefinitions of methods in derived classes. OnMessage Passing: Efficient Method Look-Up for **Dynamically** Typed Languages Jan Vitek 1 and R. Nigeland the message name (selector) Conceptually this **binding** of a message to its implementation proceeds ascul.unige.ch/OSG/people/jvitek/Publications/ecoop94.ps.gz[Axiomatizing Early and Late Input by Variable Elimination - van Deursen \(1995\) \(Correct\)](#)Axiomatizing Early and **Late** Input by Variable Elimination Arie van Deursen15 January, 1995 Abstract Variable **binding** input actions in process algebra expressions canprocess algebra, input actions can have a variable-**binding** effect, i.e. some value is read and "assigned"www.win.tue.nl:8080/cs/csr/csr95/95.21.ps[M-RPC: A Remote Procedure Call Service for Mobile Clients - Bakre, Badrinath \(1995\) \(Correct\) \(5 citations\)](#)to handle switchable server **bindings** are described **later**. 7. The agent at the new MSR starts a workermobile applications will also need the ability to **dynamically bind** mobile clients to local informationwill also need the ability to **dynamically bind** mobile clients to local information servers. Inpaul.rutgers.edu/pub/badri/mrpc.ps.Z[A Basic Model of Typed Components - Seco, Caires \(2000\) \(Correct\) \(14 citations\)](#)**dynamic binding** and subtype polymorphism, **late (dynamic)** composition, and avoidance of

ingredients like explicit context dependence, **dynamic binding** and subtype polymorphism, **late (dynamic)**
 | like explicit context dependence, **dynamic binding** and subtype polymorphism, **late (dynamic)**
ctp.di.fct.unl.pt/~jcs/papers/ecoop2000.ps.gz

Method Schemas - Abiteboul, Kanellakis, Ramaswamy.. (1994) (Correct) (8 citations)
 methods, inheritance, name overloading, and **late binding**. An important problem is to check whether
 This indicates that efficient parallel and/or **dynamic** algorithms for this problem may be hard to
 methods, inheritance, name overloading, and **late binding**. An important problem is to check whether a
wilma.cs.brown.edu/people/pck/Papers/schemas.ps

A Configurable Protocol Architecture for CORBA Environments - Crane, Dulay (1997) (Correct) (6 citations)
 is two-way. A client issues a request and at some **later** time, a server returns one or more replies. Since
 ability to select the elements of a protocol stack **dynamically** at **bind**-time depending on the properties of
 the elements of a protocol stack **dynamically** at **bind**-time depending on the properties of the interface
scoreh.doc.ic.ac.uk/~jsc/research/isads-97.ps.gz

Actuability of Underactuated Manipulators - Lee, Xu (1994) (Correct)
 during part or all of the motion [3] 1]The **late** movie star Bruce Lee, through his precise control
 the resulting underactuated system can make use of **dynamic** coupling within the system for position control.
pecan.srv.cs.cmu.edu/afs/cs.cmu.edu/user/chrislee/www/cmu-ri-tr-94-13.ps.gz

Method Resolution and Virtual Classes in a Deductive.. - Freitag (1995) (Correct)
 and propose a way to handle inheritance and **late binding** that also applies to virtual classes. Our
 [3]More specifically, 29] emphasizes that **dynamic** subclasses and class migration are essential
 and propose a way to handle inheritance and **late binding** that also applies to virtual classes. Our model
www.uni-passau.de/lehrestuehle/freitag/publications/Fr95/paper.ps

Mobile IP - Perkins, Myles, Watson (1992) (Correct) (28 citations)
 Perkins Andrew F. Myles T.J. Watson Research Center, IBM Macquarie University, Australia This paper
 cells it may migrate from cell to cell based on **dynamic** factors such as load and noise. The new
 home address and visiting address is known as a **binding**. The set of MHs using a particular IAP must be
ftp.mpce.mq.edu.au/pub/elec/dist/mobile/mip/its92.ps.Z

Mechanisms and Interfaces for Software-Extended Coherent Shared.. - Chaiken (1994) (Correct) (4 citations)
 implemented by Mathews Cherian with Kimming So at IBM. It was extended by Kiyoshi Kurihara, who found
 systems exhibit little sensitivity to trap **latency** and memorysystem code efficiency, as long as
ftp.cag.lcs.mit.edu/pub/papers/chaiken-dissert-1-10.ps.Z

Some Reflections on Computer Engineering: 30 Years after the IBM.. - Flynn (Correct)
 on Computer Engineering: 30 Years after the IBM System 360 Model 91 Michael J. Flynn 30th
 pipelined high speed machine. It was begun in the **late** 1950's and delivered in the early 1960's. It
umunhum.stanford.edu/tr/micro30.ps

Symbolic Weak Bisimulation for Value-passing Calculi - Kwak, Choi, Lee, Philippou (1998) (Correct)
 are built on the basis of a new formulation of **late** weak bisimulation which we propose. Unlike the
www.cis.upenn.edu/~lee/99cis642/99cis642/papers/MS-CIS-98-22.ps.Z

First 20 documents [Next 20](#)

Try your query at: [Google \(CiteSeer\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer - Copyright [NEC](#) and [IST](#)